Towards Client Side HTML Security Policies

Joel Weinberger, Adam Barth, Dawn Song
MySpace
Samy Worm

but most of all, samy is my hero.
Content Injection

The insertion of untrusted data, structure, or code into an application
Key Points

• Explicit policies form a compelling, unique point in the content injection protection design space

• The current trade-offs in explicit policy systems make none of the current systems completely viable

• Explicit policies are the way forward, but we need new system designs
<html>
  <h1>Forum Post #1</h1>
  <div>
    This is the content of the post.
  </div>
</html>
Content Injection

<html>
  <h1>Forum Post #1</h1>
  <div>
    <script>alert(document.cookie);</script>
  </div>
</html>
<html>
  <h1>Forum Post #1</h1>
  <div>
    <script>alert(document.cookie);</script>
  </div>
</html>
Web Application Frameworks

- Systems for writing web applications
- Frameworks provide tools for sanitizing content

- Turns out, sanitization is hard
  - Shameless plug for our ESORICS 2011 paper:

  *A Systematic Analysis of XSS Sanitization in Web Application Frameworks*
Implicit Policies

Browser

Web Application Policy
Explicit Policies

Browser

Web Application

Policy
Key Points

✓ Explicit policies form a compelling, unique point in the content injection protection design space

• The current trade-offs in explicit policy systems make none of the current systems completely viable

• Explicit policies are the way forward, but we need new system designs
Explicit Policy Systems

- BEEP
- BLUEPRINT
- Content Security Policy (CSP)
BEEP

• Hashes of allowed scripts

• Performance: good

• Dynamic scripts are very hard to get right

• Only XSS
BLUEPRINT

• Structural description of page, enforced by JavaScript library

• Performance: poor

• Does not trust the browser’s parser

• Very fine grained granularity
Content Security Policy (CSP)

- Specify allowed behaviors of page

- Performance: ?

- Only handles some content injection

- Coarse grained
  - What is the affect on how applications are written?
Applying CSP to Applications

- How does CSP affect Web applications?
- Apply CSP to Bugzilla and HotCRP
- Measure performance of applications and how the applications were changed
CSP Study

• Developer effort to retrofit applications to be CSP compatible is large

• Template variables cannot be used in scripts

• Need to lookup data through JavaScript

• Template logic no longer affects scripts
## CSP Study

### Bugzilla

<table>
<thead>
<tr>
<th>Page</th>
<th>No Inline JS</th>
<th>Async JS</th>
</tr>
</thead>
<tbody>
<tr>
<td>index.cgi</td>
<td>14.8%</td>
<td>-3.0%</td>
</tr>
<tr>
<td>editsettings.cgi</td>
<td>6.3%</td>
<td>5.1%</td>
</tr>
<tr>
<td>enter_bug.cgi</td>
<td>57.6%</td>
<td>44.2%</td>
</tr>
<tr>
<td>show_bug.cgi</td>
<td>51.5%</td>
<td>4.0%</td>
</tr>
</tbody>
</table>

### HotCRP

<table>
<thead>
<tr>
<th>Page</th>
<th>No Inline JS</th>
<th>Async JS</th>
</tr>
</thead>
<tbody>
<tr>
<td>index.php</td>
<td>45.3%</td>
<td>37.2%</td>
</tr>
<tr>
<td>search.php</td>
<td>52.9%</td>
<td>50.4%</td>
</tr>
<tr>
<td>settings.php</td>
<td>23.3%</td>
<td>16.1%</td>
</tr>
<tr>
<td>paper.php</td>
<td>61.1%</td>
<td>58.5%</td>
</tr>
<tr>
<td>contacts.php</td>
<td>67.8%</td>
<td>35.5%</td>
</tr>
</tbody>
</table>
Key Points

✓ Explicit policies form a compelling, unique point in the content injection protection design space

✓ The current trade-offs in explicit policy systems make none of the current systems completely viable

• Explicit policies are the way forward, but we need new system designs
Explicit Policies: The Good and the Bad

• Provide a separation policy from application
  • Not doing this makes security hard

• Simple or complex: you choose

• Not good at performance and developer usability
Towards the Future

• Policy systems are useful and should be how we approach content injection

• CSP has some great properties, but suffers when applied to current applications

• How can we combine features from these different systems?
Key Points

- Explicit policies form a compelling, unique point in the content injection protection design space

- The current trade-offs in explicit policy systems make none of the current systems completely viable

- Explicit policies are the way forward, but we need new system designs